Using GPG to Encrypt Your Data

Encryption helps protect your files during inter-host file transfers that use protocols that are not already encryptedâ for example, when using **ftp** or when using **shiftc** without the **--secure** option. We recommend using the GNU Privacy Guard (GPG), an Open Source OpenPGP-compatible encryption system.

GPG has been installed on Pleiades, Endeavour, and Lou in the /usr/bin/gpg directory. If you do not have GPG installed on the system(s) that you would like to use for transferring files, please see the GPG website.

Choosing What Cipher to Use

We recommend using the cipher AES256, which uses a 256-bit Advanced Encryption Standard (AES) key to encrypt the data. Information on AES can be found at the National Institute of Standards and Technology's Computer Security Resource Center.

You can set your cipher in one of the following ways:

- Add **--cipher-algo AES256** to your ~/.gnupg/gpg.conf file.
- Add **--cipher-algo AES256** in the command line to override the default cipher, CAST5.

Examples

If you choose not to add the **cipher-algo AES256** to your gpg.conf file, you can add **--cipher-algo AES256** on any of these simple example command lines to override the default cipher, CAST5.

Creating an Encrypted File

Both commands below are identical. They encrypt the test.out file and produce the encrypted version in the test.gpg file:

```
% gpg --output test.gpg --symmetric test.out

% gpg -o test.gpg -c test.out
```

You will be prompted for a passphrase, which will be used later to decrypt the file.

Decrypting a File

The following command decrypts the test.gpg file and produces the test.out file:

```
% gpg --output test.out -d test.gpg
```

You will be prompted for the passphrase that you used to encrypt the file. If you don't use the **--output** option, the command output goes to STDOUT. If you don't use any flags, it will decrypt to a file without the .gpg suffix. For example, using the following command line would result in the decrypted data in a file named "test":

```
% gpg test.gpg
```

Selecting a Passphrase

Your passphrase should have sufficient information entropy. We suggest that you include five words of 5-10 letters in size, chosen at random, with spaces, special characters, and/or numbers embedded into the words.

You need to be able to recall the passphrase that was used to encrypt the file.

Factors that Affect Encrypt/Decrypt Speed on NAS Filesystems

We do not recommend using the **`--armour`** option for encrypting files that will be transferred to/from NAS systems. This option is mainly intended for sending binary data through email, not via transfer commands such as **`ftp`** or **`shiftc`** with the **`-secure`** option. The file size tends to be about 33% bigger than without this option, and encrypting the data takes about 10-15% longer.

The level of compression used when encrypting/decrypting affects the time required to complete the operation. There are three options for the compression algorithm: **`none`**, **`zip`**, and **`zlib`**.

- **`--compress-algo none`** or **`--compress-algo 0`**
- **`--compress-algo zip`** or **`--compress-algo 1`**
- **`--compress-algo zlib`** or **`--compress-algo 2`**

For example:

```
% gpg --output test.gpg --compress-algo zlib --symmetric test.out
```

If your data is not compressible, **`--compress-algo 0`** (**`none`**) gives you a performance increase of about 50% compared to **`--compress-algo 1`** or **`--compress-algo 2`**.

If your data is highly compressible, choosing the **`zlib`** or **`zip`** option will not only increase the speed by 20-50%, it will also reduce the file size by up to 20x. For example, in one test on a NAS system, a 517 megabyte (MB) highly compressible file was compressed to 30 MB.

The **`zlib`** option is not compatible with PGP 6.*x*, but neither is the cipher algorithm AES256. Using the **`zlib`** option is about 10% faster than using the **`zip`** option on a NAS system, and **`zlib`** compresses about 10% better than **`zip`**.

Random Benchmark Data

We tested the encryption/decryption speed of three different files (1 MB, 150 MB, and 517 MB) on NAS systems. The file used for the 1 MB test was an RPM file, presumably already compressed, since the resulting file sizes for the **`none/zip/zlib`** options were within 1% of each other. The 150 MB file was an ISO file, also assumed to be a compressed binary file for the same reasons. The 517 MB file was a text file. These runs were performed on a CXFS filesystem when many other users' jobs were running. The performance reported here is for reference only, and not the best or worst performance you can expect.

Using AES256 as the Cipher Algorithm			
	1 MB File	**150 MB File**	**517 MB File**

with --armour	~5.5 secs to encrypt	~40 secs to encrypt	
without --armour	~4 secs to encrypt	~35 secs to encrypt	
without --armour, zlib compression		~33 secs to encrypt; ~28 secs to decrypt to file	~33 secs, resultant file size ~30 MB; ~34 secs to decrypt to file
without --armour, zip compression		~36 secs to encrypt; ~31 secs to decrypt to file	~38 secs, resultant file size ~33 MB; ~34 secs to decrypt to file
without --armour, no compression		~19 secs to encrypt; ~25 secs to decrypt to file	~49 secs, resultant file size ~517 MB; ~75 secs to decrypt to file

---

Article ID: 242
Last updated: 28 Apr, 2022
Revision: 15
Transferring Files & Data -> Remote Transfers -> Using GPG to Encrypt Your Data
https://www.nas.nasa.gov/hecc/support/kb/entry/242/